Short tutorial on STL tools and functions used in Siconos

F. Pérignon

For Kernel version 1.1.4
May 24, 2006

1 Introduction

This paragraph is supposed to give to Siconos users a minimal knowledge on how to handle the STL functions and objects that are used in Siconos and thus necessary to built properly a Non Smooth Dynamical System or a Strategy.
For more details on the STL tools, see for example *http://www.sgi.com/tech/stl/table_of_contents.html* or *http://cplus.about.com/od/stl/*.
Three main objects are widely used in Siconos:

- map<key,object>: object that maps a "key" of any type to an "object" of any type.
- set<object>: a set of objects ...
- vector<object>: a list of objects, with a specific sorting; data can also be accessed thanks to indices. Moreover, objects handled by a vector are contiguous in memory.

2 Iterators and find function

Iterators are a generalization of pointers: they are objects that point to other objects. They are used to iterate over a range of objects. For example, if an iterator points to one element in a range, then it is possible to increment it so that it points to the next element.
For example, suppose you have a set<DynamicalSystem*>, then you can define an iterator in the following way:
set<DynamicalSystem*>::iterator iter ;
iter will be used to access data in the set.

It is not necessary to give more details about iterators in that paragraph. Mainly, in Siconos users .cpp input files, they will only be used as a return value for find function. Then just used them as described in examples for map or set in the paragraphs below.

3 Map handling

Let A be a map<string,DynamicalSystem*>.

Data access:

- A[name] is the DynamicalSystem* that corresponds to the string name. Thus to fill a map in, just used A[name] = B, with B a DynamicalSystem*.
- A.erase(name) : (name of type string) erases the element whose key is name.
- A.clear(): removes all elements in the map.

- A.size() : number of elements in the map.
- A.find(name) or A.find(DS), name a string and DS a DynamicalSystem*: returns an iterator that points to DS. See example below.

Example:

```
// creates a map of DynamicalSystem*

map<string, DynamicalSystem*> A;

DynamicalSystem * DS = new DynamicalSystem(...);

// Add some DynamicalSystem* in the map
A["FirstDS"] = DS;
A["SecondDS"] = new DynamicalSystem(...);
...

map<string,DynamicalSystem*>::iterator iter;

// Find a DynamicalSystem* in the map
iter = A.find(DS);

// Then iter points to DS
(*iter)→display() ; // display data of DS

DynamicalSystem * DS2 = new DynamicalSystem(...);
iter = A.find(DS2);

// In that case, since DS2 is not in the map,
// iter is equal to A.end();
// It is then easy to test if a DynamicalSystem* is in the map or not.

delete DS;
delete DS2;
delete A["secondDS"];
A.clear();
```

4 Set handling

Let A<DynamicalSystem*> be a set.

Data access (DS being a DynamicalSystem*):

- A.insert(DS) adds DS into the set
- A.erase(DS) removes DS from the set
- A.find(DS) returns an iterator that points to DS. See example below.

Example:

```
// creates a set of DynamicalSystem*
set<DynamicalSystem*> A;

DynamicalSystem* DS = new DynamicalSystem(...);
DynamicalSystem* DS2 = new DynamicalSystem(...);

// add elements into the set
A.insert(DS);
A.insert(DS2);

A.size(); // is equal to 2.

// find an element:
set<DynamicalSystem*>::iterator iter;
iter = A.find(DS);

// then iter points to DS;
(*iter)→display(); // display DS data.

// remove an element
A.remove(DS2);
// A.size() is then equal to 1.

iter = A.find(DS2);
// then iter = A.end(), which means that DS2 is not in the set anymore.

delete DS;
delete DS2;
```

5 Vectors handling

Let V<DynamicalSystem*> be a vector.

Data access:

- V[i] is the component at position i in the vector, and so is a DynamicalSystem*.
- V.size(): number of elements in V.
- V.push_back(DS) : adds the DynamicalSystem* DS at the end of V.
- V.pop_back() : removes the last element of V.
- V.clear() : removes all elements.

Examples:

```
// Creates a vector of three elements, that contains DynamicalSystem*
vector<DynamicalSystem*> V(3);
V[0] = new DynamicalSystem(...);

DynamicalSystem* DS = new DynamicalSystem(...);
V[1] = DS;

V[2] = NULL;
```

```
// At this points, V.size() is equal to 3.

DynamicalSystem∗ DS2 = new DynamicalSystem(...);

V.push_back(DS2);

// then V.size() is equal to 4.

// DS display:
V[1]→display();

...

delete DS2;
delete DS;
delete V[0];
V.clear();
```